Technical Report (20.12.2019)

Brute-force calculation of aperture diffraction in camera lenses

Emanuel Schrade, Johannes Hanika and Carsten Dachsbacher

Institute for Visualization and Data Analysis (IVD)
Karlsruhe Institute of Technology, Germany

Abstract

Sensorrealistic image synthesis requires precise simulation of photographic lenses used in the imaging process. This is well understood using geometric optics, by tracing rays through the lens system. However, diffraction at the aperture results in interesting though subtle details as well as prominent glare streaks and starry point spread functions. Previous works in graphics have used analytical approximations for diffraction at the aperture, but these are not well suited for a combination with distortion caused by lens elements between the aperture and the sensor. Instead we propose to directly simulate Huygens' principle and track optical path differences, which has been considered infeasible due to the high computational demand as Monte Carlo simulation exhibits high variance in interference computations due to negative contributions. To this end we present a simple Monte Carlo technique to compute camera point spread functions including diffraction effects as well as distortion of the diffracted light fields by lens elements before and after the aperture. The core of our technique is a ray tracing-based Monte Carlo integration which considers the optical path length to compute interference patterns on a hyperspectral frame buffer. To speed up computation, we approximate phase-dependent, spectral light field transformations by polynomial expansions. We cache transmittance and optical path lengths at the aperture plane, and from there trace rays for spherical waves emanating to the sensor. We show that our results are in accordance with the analytical results both for near and far field.

1. Introduction

In recent years, the field of photorealistic image synthesis has advanced such that we are now able to compute visually rich imagery that is almost indistinguishable from real photographic footage. For seamless integration of computer generated imagery (CGI) into real world photography, the lens distortions are either corrected out of the plate or simulated in CGI. To achieve the particular indistinguishable look of certain classic lenses, however, the optical system needs to be simulated as closely as possible. Consider for instance the characteristic depth of field in Kubrick's *Barry Lyndon*, due to the T0.7 aperture, or the signature lens flares of old movies like *Dirty Harry* due to the coated anamorphic lenses. To replicate these effects in a renderer with high precision, we need to consider diffraction at the aperture blades. A second important aspect is that diffraction limits the size of the point spread function and hence the sharpness of the image.

Computing diffraction patterns is a hard problem. One approach is to use analytical approximations, which, however, require making certain assumptions, e.g. that the incident illumination is constant across the aperture, is a simple orthogonal plane wave, or that the exitant radiance is only needed in the near field or far field. None of these assumptions hold in our context where illumination comes from the scene and is distorted by a few lens elements before and after it passes through the aperture.

As solution to this problem, we propose a technique based on simulating the spherical waves in Huygens' principle directly. Essentially, we stop tracing a light transport path when it passes through the aperture opening. At this point, we continue by tracing in a random new direction to sample the spherical wave emanating from there. We also track the optical path difference while continuing the path up to the sensor. At the sensor a spectral frame buffer stores the amplitude and phase of an incoming path and accounts for interference with other paths of the same wavelength.

This approach is typically considered being computationally intractable as the variance of a Monte Carlo estimator easily becomes unbounded in the presence of interference (due to negative contributions) and due to the large number of samples required for converged results with high-frequency interference patterns.

We describe the following steps which nonetheless make this approach feasible: first, we collapse the ray tracing through the lens system by expressing the transformation of the light field with a polynomial [SHD16]. We extend these polynomials to additionally compute the optical path length for a path through the lens. Second, we observe that changing the direction of the path at the aperture effectively makes the transport encountered after the aperture independent of that before the aperture. We can thus cache the optical path difference and transmittance values at the aperture and decorrelate the computation before and after the aperture. Lastly, we present an efficient GPU implementation.

Altogether, these contributions make it possible to efficiently

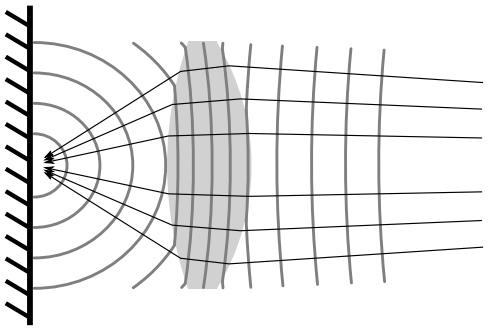


Figure 1: *Spherical wavefronts emitted by the light source on the right side propagate towards the biconvex lens. Waves are slowed down when entering the glass. Thanks to the shape and material of the lens the wavefronts converge towards a point on the sensor. The same behaviour can be observed in geometric optics by applying Snell's law. Note, that the rays are always orthogonal to the wavefronts, that is, the rays coincide with the propagation direction of the wavefronts.*

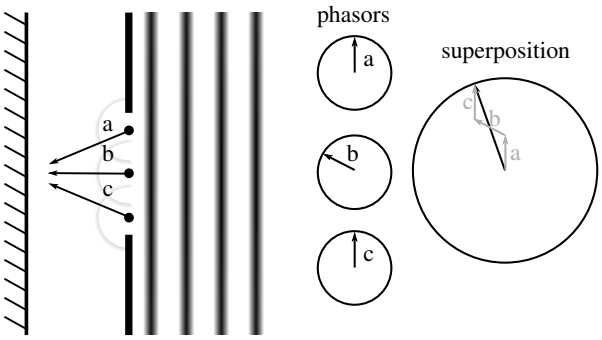


Figure 2: *An aperture is illuminated by a plane wave. Using Huygens' principle, points on the aperture can be treated as sources of elementary waves that interfere with each other on the sensor. The relative phase of a wave when reaching a pixel on the sensor can be incorporated into a phasor. By accumulating these phasors we obtain a new phasor describing the superposition of the incoming waves. Here the length of* a *and* c *is approximately* 3λ *while* b *has a length of approximately* 2.8λ

compute a point spread function for sensor-realistic Bokeh. As an outlook, we also show how to compute lens flares with diffraction effects, and point out which changes will be necessary in the future to make this faster.

2. Background and Previous Work

Wave optics In wave optics, light is modeled as light waves emerging from sources and propagating through the scene (e.g. see Born et al. [BWB*99] for an introduction). One source of diffraction patterns on the sensor is the superposition of light waves from the scene. To calculate the superposition in a point so-called *phasors* can be used. A phasor stores the phase and intensity of a light wave for a specific frequency as a complex number. The resulting intensity of the superposition of waves can be calculated as the amplitude of the phasor describing the superimposed wave which is simply the sum of the original waves' phasors. Whether waves interfere constructively or destructively – resulting in an intensity maximum or minimum, respectively – depends on the phase of the interfering waves. In free space the phase can be calculated directly from the distance travelled. For a light wave emitted in a point $\mathbf{l}$ with wavelength λ and initial phase $\varphi(\mathbf{l}) = 0$ the phase in $\mathbf{x}$ is

$$\varphi(\mathbf{x}) = \frac{2\pi}{\lambda}|\mathbf{x}-\mathbf{l}|.$$

Points with an equal phase form a *wavefront*; for a point light source the wavefronts form spheres centered at the light source. In Figure 1 we show an example for such wavefronts propagating towards the sensor. When propagating through materials we need to account for their refractive indices η to calculate a wave's phase in a point using the *optical path length* (*OPL*):

$$OPL = \int_{path} \eta \, ds, \text{ and hence } \varphi = \frac{2\pi}{\lambda} \int_{path} \eta \, ds.$$

To model diffraction caused by the lens aperture, we use *Huygens' principle* which states that each point on a wavefront acts as a source of a new spherical elementary wave. By tracing these waves to the sensor and accumulating their phasors we calculate the diffraction pattern. Figure 2 shows an example where phasors of elementary waves are accumulated on the sensor to calculate the diffraction pattern of a single slit.

Approximations Because of the enormous effort in computing such diffraction effects, various approximations for specific scenarios have been proposed. Assuming that the diffracted light field is only interesting in the far field, there exists the Fraunhofer approximation. Conversely, there is a near field approximation called Fresnel diffraction. If only the most prominent maxima are needed, diffraction effects can be approximated by geometric optics [Kel62]. Another approach to combine geometric optics with diffraction is to analytically describe the diffracted light field by a Wigner distribution and sample emanating rays from this [Alo11].

Diffraction in lens design Commercial lens design software packages such as *ZEMAX* or *CODE V* support the simulation of advanced diffraction effects. The methods applied in the latter are apparently based on beamlet tracing, i.e. similar to Harvey and Pfister [HIP15] or Mout et al. [MWB*16]. This work also provides a good background on the state of the art in diffraction simulation, e.g. based on plane wave decomposition, and the drawbacks of such approximations. Their work is based on decomposing a wave into Gaussian beams of a certain width. If the right density of these primitives is combined, the result can be seamlessly reconstructed. Our work is much simpler and based on classical ray tracing. Further we are interested in calculating diffraction patterns for different wavelengths and covering large parts of the sensor, not only a small area around the focus point on the sensor.

Diffraction in graphics In computer graphics diffraction effects have been simulated for a few special cases only. For instance using *diffraction shaders* for snake skin in the far field [Sta99]. More

recent works on far field diffraction at surface interactions, such as microscratches [YHW*18], still assume uniform incoming light.

More closely related to our work, diffraction at lens apertures has been approximated using the fractional Fourier transform [PF94] as continuum between near and far field approximations for realistic lens flares [HESL11]. Computation of lens diffraction can be accelerated by assuming constant incident illumination [SLE18] based on analytical integration over quads. However, this work assumes that the diffracted light field is not further distorted by any lens elements. The geometric theory of diffraction has also been explored to create starry aperture diffraction effects [SDHL11], but is inaccurate for Bokeh rendering in the near field.

Lens simulation with geometric optics Realistic lens models have been explored in computer graphics by considering image distortions due to thick lenses [KMH95], and through ray tracing of lens descriptions [SDHL11]. To speed up the computation for lens flares, where the paths through the lens system can get long, ray tracing has been approximated by polynomials, using a Taylor expansion of the light field transformation [HHH12]. This has also been used to synthesize Bokeh in combination with efficient importance sampling for small aperture openings [HD14]. This line of work has been made more accurate by replacing the Taylor expansion by a fitting process to better match the results far away from the optical axis [SHD16]. This improves support for aspheric elements and makes the methods more suitable for wide angle lenses. To speed up realistic Bokeh computation, the point spread function (PSF) has also been precomputed into Bokeh textures [JKL*16].

We do not precompute individual textures, as they vary with incident light direction. As previous work, we employ polynomials from the light source to the aperture, and from the aperture to the sensor. However, we change the parametrization and compute additional polynomials for transmission and optical path length. As we split the paths where they pass through the aperture, we do not require iterative importance sampling techniques to find a specific point on the opening in contrast to previous work.

3. Algorithm

In the following we describe our new approach to calculate diffraction effects. First we demonstrate for a simple example that our algorithm agrees with the analytic approximation that is often used in physics for this special case. Then we move on to calculating diffraction patterns caused by a lens aperture.

3.1. Diffraction by a single slit in 1D

For a single one-dimensional slit that is illuminated by a plane wave, such as in Figure 2, each point on the slit acts as a source of a new spherical light wave (Huygens' principle). The assumption of incoming plane waves simplifies the calculations as each elementary wave has the same initial phase. The intensity in a point on the sensor is the sum of the phasors for all elementary waves:

$$I(\mathbf{s}) = \frac{1}{\lambda}\int_{-r}^{r} \frac{e^{i\frac{2\pi}{\lambda}|\mathbf{s}-\mathbf{a}|}}{|\mathbf{s}-\mathbf{a}|}\cdot\cos^2\alpha \;\mathrm{d}\mathbf{a}$$

Algorithm 1 Slit Diffraction

procedure PIXELINTENSITY(y,λ)
Input: y: position on sensor, λ: wavelength,
    d: distance from aperture to sensor
Output: Intensity on pixel
    $I \leftarrow (0,0)$
    for $i \leftarrow 1 \dots N$ **do**
        $y_{\text{ap}} \leftarrow$ random_aperture_point()
        $\text{OPL} \leftarrow \sqrt{(y-y_{\text{ap}})^2+d^2}$
        $\varphi \leftarrow \frac{2\pi}{\lambda}\text{OPL}$
        $I \leftarrow I + \sqrt{\frac{d^2}{\lambda \text{OPL}^3}}(\cos\varphi, \sin\varphi)$
    return $\frac{|I|^2}{N}$

where $\mathbf{a}$ are points on the slit with half slit width r we integrate over, $\mathbf{s}$ is the point on the sensor, and α is the angle between the aperture or sensor normal and the direction from the aperture point to the sensor point. Often it is further assumed that the screen is far enough from the aperture such that all lines connecting $\mathbf{s}$ and $\mathbf{a}$ are parallel; this allows for a closed-form solution for the intensity on the screen:

$$I(\mathbf{s}) \propto \operatorname{sinc}\left(\frac{2\pi r \sin\alpha}{\lambda}\right),$$

for wavelength λ and α being the angle between the optical axis and the direction of the ray from the aperture point $\mathbf{a}$ to the sensor point $\mathbf{s}$. In preparation for the next step, we evaluate the diffraction numerically using Monte Carlo integration without the need for further assumptions; the approach is summarized in Algorithm 1. Note that this algorithm can be directly applied for any sensor distance and for any aperture, e.g. it works directly for calculating interference in a double slit experiment. In Figure 3 we show that our integration agrees with analytic approximations. We are also able to handle incoming waves on the aperture that are not planar by simply adding their initial phase which is shown in Figure 3 for converging wavefronts as they occur in lens systems (see Figure 4).

3.2. Diffraction in camera lenses

Since we focus on diffraction caused by the lens aperture in this work, the light transport in the scene remains unaffected. On the sensor we accumulate phasors in a complex framebuffer similar to the one-dimensional example shown before. We first calculate the transport from the scene to the aperture, then apply Huygens' principle to account for diffraction effects caused by the aperture, and afterwards calculate the transport to the sensor.

For accumulating light from the scene on the sensor we compute the transport between the scene point and random points on the aperture. We calculate the phase at each aperture point using a polynomial for the OPL. The transmittance is calculated from the incoming radiance and the transmittance through the lens, which is again described by a polynomial.

On the aperture we can effectively choose the outgoing direction to the sensor freely, as in the previous example. We can thus trace only the parts of the elementary wave that contribute to the

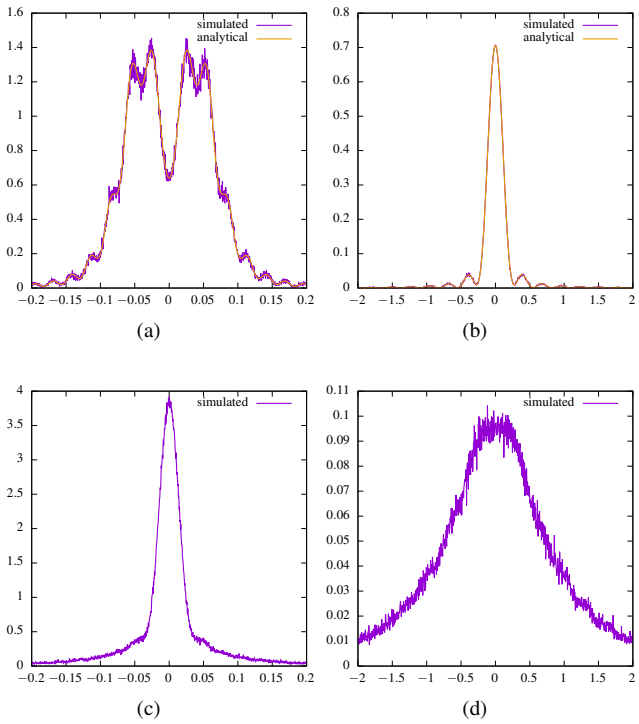


Figure 3: *For incoming plane-waves integration with our approach agrees with the analytical solutions generally used in optics for both the near-field (a) and the far-field (b). For other phase distributions the analytical solution cannot be used directly. In (c) and (d) we changed the incoming wave on the aperture to be converging as it is the case in the lens system.*

result. Specifically we are interested in the phase and amplitude in the pixel centers on the sensor where we simply accumulate the phasors in a complex frame buffer. The final pixel intensity can be calculated as the squared amplitude of the pixel's value in the framebuffer.

The phasors for each contributing wave are defined by the optical path length of the (bent) ray through the lens system and the radiance arriving at the sensor. The radiance from the scene is attenuated by the transmittance through the lens system, by the cosine of the angle of the ray incident to the aperture plane, and the cosine of the ray exiting the aperture plane. As the light now propagates through several lens elements with different refractive indices before and after passing through the aperture, the optical path length is more complex than before. In Figure 4 we show the relative phase and the transmittance over the aperture for rays going through one point in the scene. The phase and transmittance for a ray passing through a given point on the aperture can be obtained by simply evaluating the corresponding polynomials.

The spherical wavefronts emanating at the aperture transport light in every direction which allows us to splat each sample to all pixels on the sensor that can be reached by the aperture point. This observation allows for a decoupling of the light transport between the scene and the aperture on the one hand, and between the aperture and the sensor on the other hand.

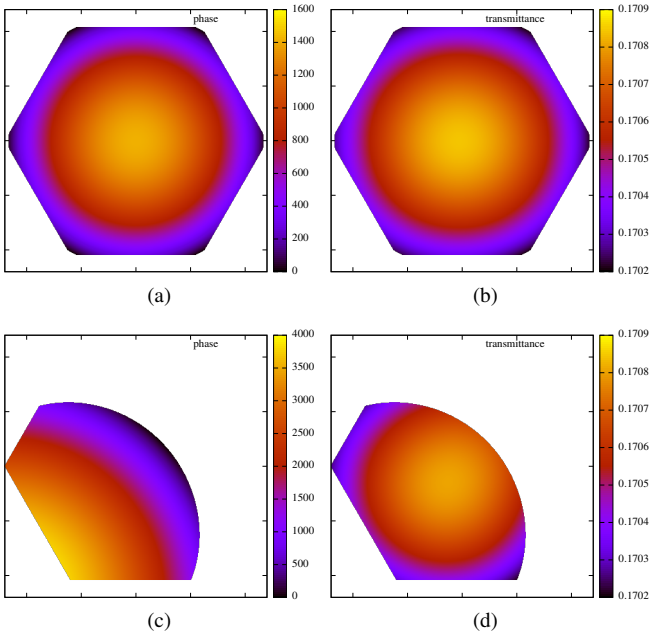


Figure 4: *Relative phase and transmittance over points on the aperture for a light source on the optical axis (a), (b) and off-axis (c), (d). While the transmittance can be considered almost constant, the phase varies by 1600 and 4000 whole waves between points on the aperture.*

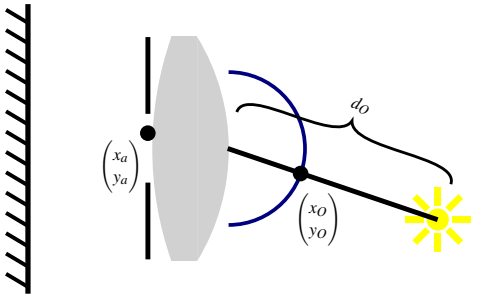


Figure 5: *We parameterize a path by the distance and direction of the light source, the wavelength and the aperture position. The polynomial outputs the position on the outer pupil for clipping and visibility testing, the transmittance and the optical path length.*

3.3. Light field transformation through polynomials

Polynomials were used before to calculate the transport of rays through a lens system for rendering [SHD16]. In this context, however, care needs to be taken to sample a point on the (potentially small) aperture. This is because both the direction and the position at the aperture are completely determined by the point in the scene. While this is still the case for us, we trace into all relevant directions

of the spherical wave after passing the aperture. Hence, instead of performing Newton iterations to obtain a ray passing through a sampled point on the aperture or in the scene, we notice that the phase and transmittance for one point in the scene change smoothly over the area of the aperture (see Figure 4). When moving the point in the scene, the change is also smooth. This observation leads us to a new parametrization for the polynomial: We fit polynomials that describe the phase and transmittance for the transport between a point on the aperture and one in the scene. We also use the orthogonal matching pursuit algorithm from previous work [SHD16, Alg. 1] to reduce the number of terms in the polynomials. To determine visibility in the scene and clipping by the outer pupil of the lens we further fit polynomials to calculate the ray position on the outer pupil. The following 6×4 polynomial system describes the transport between the aperture and scene:

$$P_o(\mathbf{O}) : (x_O, y_O, d_O, x_a, y_a, \lambda) \mapsto (x_o, y_o, \tau_o, \varphi_o), \tag{1}$$

where $d_O = |L|$ is the distance from the scene point in camera coordinates to the center of the outer pupil, $x_O = x_L/d_O$ and $y_O = y_L/d_O$ are the position projected onto the unit hemisphere (see Figure 5), λ is the wavelength, and $x_a,\ y_a$ are the coordinates of the aperture point. Additionally to visibility calculation, we use the position on the outer pupil $x_o,\ y_o$ to clip rays on the outer pupil.

This polynomial system allows us to transport light from the scene to the aperture and we use the same approach for transporting it to the sensor. Note again that we use the fact that due to diffraction, we can change the direction on the aperture arbitrarily, hence this approach is not applicable in a general path tracing framework. For the transport to the sensor we have

$$P_s(\mathbf{S}) : (x_S, y_S, z_S, x_a, y_a, \lambda) \mapsto (x_i, y_i, \tau_i, \varphi_i), \tag{2}$$

where $x_S,\ y_S$ define the point on the sensor, and z_S can be used as a sensor offset for focusing. Here $x_i,\ y_i$ are points on the inner pupil for clipping. The need for clipping rays at the inner and outer pupil can be seen in Figure 4 especially for the case of an off-axis light source. In the results section we show rendered point spread functions also for the case where it is clipped by the outer pupil.

The point spread function for a fixed point in the scene and a fixed wavelength can be calculated for each pixel by simply integrating over the aperture:

$$I = \int_{\mathbf{x_a} \in \mathrm{A}} \tau_o(\mathbf{x_a}) \tau_i(\mathbf{x_a}) \frac{e^{i \frac{2\pi}{\lambda} (\varphi_o(\mathbf{x_a}) + \varphi_i(\mathbf{x_a}))}}{\varphi_o(\mathbf{x_a}) + \varphi_i(\mathbf{x_a})} \, \mathrm{d}a,$$

which we perform by standard Monte Carlo integration, sampling $\mathbf{x_a}$ on the aperture and accumulating phasors in a spectral framebuffer.

3.4. Extension for lens flares

The polynomials can as well be fitted and used for rendering lens flares. However, we noticed artifacts due to unclipped rays that would not pass the lens such as in Figure 6. As a consequence the flares are generally too large and too much light reaches the sensor due to the description of the transformation through polynomials. To alleviate this problem we added two more polynomials for clipping lens flares

$$P_{\text{clip}} : (x_O, y_O, d_O, x_a, y_a, \lambda) \mapsto (x_{\text{clip}}, y_{\text{clip}}). \tag{3}$$

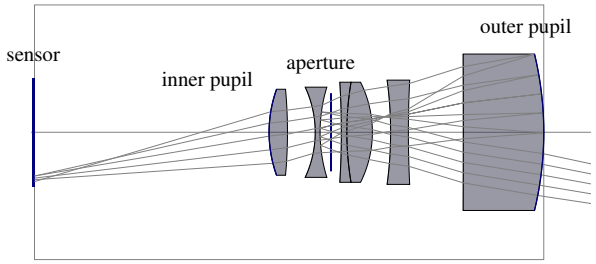


Figure 6: *Lens flare in an anamorphic lens (tessar-anamorphic). Some rays at the upper part of the ray bundle would not be clipped at the inner pupil, aperture, or outer pupil. Additional clipping at the reflection at the outer pupil is necessary for a correct result.*

x_{clip} and y_{clip} define the maximum distance to the lens center relative to the housing radius. For the above example (see Figure 6) the distance is maximized at the reflection at the outer pupil which is where rays have to be clipped, additionally to clipping at the inner pupil, aperture, and outer pupil. We take the distance relative to the lens housing radius so that no additional radius has to be stored. Simply checking if

$$\sqrt{x_{\text{clip}}^2 + y_{\text{clip}}^2} \leq 1$$

is sufficient to decide if the ray needs to be clipped.

4. Implementation Details

We obtain polynomials for describing the light transport in the lens system by fitting the coefficients of general polynomials using the orthogonal matching pursuit algorithm akin to Schrade et al. [SHD16, Alg. 1]. We use a ray tracer to generate random rays through the lens system for fitting the polynomials. The camera lens is described by a sequence of surfaces each with a radius of curvature, thickness, and a material definition. Surfaces can also be aspheric to reduce aberrations, or cylindrical to build anamorphic lenses (often used in the film industry for a wider aspect ratio).

We trace rays from the sensor through the lens system to calculate the position on the aperture and the outer pupil. Additionally we accumulate the optical path length during ray tracing and calculate the transmittance through the Fresnel equations. The data relevant for Eqs. (1), (2) and (3) is gathered from ray tracing random rays and then fed to the fitter as in previous work.

Lens Flares. For lens flares we added the clipping polynomial described above to the fitter. The maximal relative distance to the optical axis is tracked by the ray tracer. Additionally reflections at specified lens surfaces were added and the reflectance is accounted for. We also added anti-reflection coatings as they are generally used when building lenses and add colors to the flares. As the material and the exact thickness of coatings on the different lens surfaces is usually not made public by lens manufacturers, we approximate them through a $\lambda/4$ coating of magnesium fluoride – a material which is often used for such coatings.

Polynomial evaluation. We chose the polynomial parametrization such that an efficient evaluation is possible. For the polynomial describing the transport between the scene and the aperture, only a bivariate polynomial remains after inserting the wavelength and the point in the scene. Then the phase and transmittance as well as the information for clipping rays can be evaluated directly from the aperture position, where points on the aperture can be sampled randomly. Furthermore the transport between sensor and aperture is independent from the transport to the scene, so each aperture sample can be splatted to the whole sensor, except for rays that are clipped.

We found that single-precision floating-point numbers are not sufficient for the optical path length. These values can be in the order of meters, mainly bound by the size of the scene, while we need a precision in the order of nanometres to be able to calculate the interference of waves. Hence, we use doubles for the calculation of the optical path length, and single-precision otherwise. Instead of calculating the phasors directly from the optical path length, we first calculate the relative phase $0 \leq \varphi_{\text{rel}} \leq 2\pi$ by subtracting multiples of the wavelength from the optical path length.

For rendering the point spread function for one point light and N samples per pixel, the outer polynomial needs to be evaluated N times, and the sensor polynomial N times per pixel. When considering more than a single light source, the different scene points can be accumulated on the aperture, resulting in N evaluations per scene point to calculate the phasors on the aperture; note that the transport to the sensor is unchanged and still requires N evaluations per pixel.

Phasors of different wavelengths cannot be added, hence we need a framebuffer for accumulating phasors for each wavelength. We use 16 discrete uniformly distributed wavelengths however, the number of wavelengths can be arbitrarily changed when keeping in mind that more storage and potentially more samples are needed.

We calculate the superposition of waves only at the pixel centers, as integrating over the pixel surface would mean that waves at different positions can interfere with each other, which is obviously not correct.

GPU implementation. We implemented our method on a GPU using OpenGL. It consists of two compute shaders: The first one evaluates the polynomial from the scene to sampled aperture points and stores the result in a texture. The second compute shader transports light from the aperture points to each pixel on the sensor. Samples on the aperture are generated on the CPU using the C++ Mersenne Twister pseudo random number generator.

The run time is independent of the position of the point light source or the complexity of the diffraction pattern on the sensor. Only the number of samples required for a converged image differs. In general we noticed that larger apertures require more samples to converge. The $f/2$ aperture in Figure 7 is rendered with 32 million samples per pixel, whereas the aperture in Figure 9 was rendered with less than 6 million samples per pixel. Our implementation is capable of calculating 2.65 million aperture samples per pixel per hour on an AMD Radeon R9 390 graphics card. For each sample the first polynomial is evaluated once and the sample is afterwards splatted to all 1440×1080 pixels in the frame buffer. The samples are distributed equally among 16 wavelengths and we have a two-channel floating point texture array for accumulating the phasors with one layer per wavelength.

5. Evaluation

We have seen that our numerical approach agrees with the analytic solutions, which are available for certain settings such as the one-dimensional slit as shown in Figure 3. Two-dimensional images of aperture diffraction in a camera lens can be calculated analogously through accumulating the optical path lengths of all incoming waves on the sensor, accelerated using our polynomials. Figure 7 shows the point spread function of a point light source where rays pass the lens close to the border. Even for wide opened apertures diffraction effects are visible as ringing at the aperture blades. For smaller apertures diffraction streaks occur.

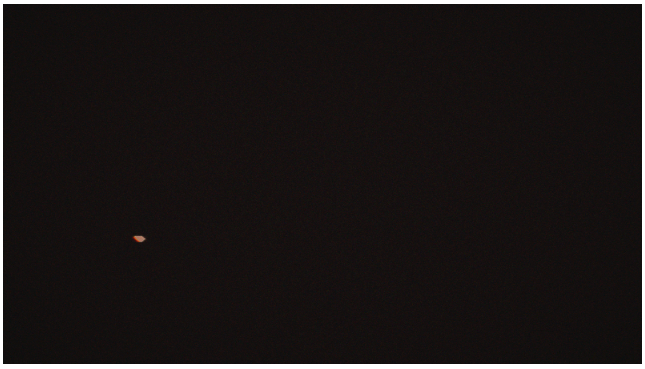
In previous work diffraction was approximated through the fractional Fourier transform of the aperture shape. In Figure 8 we show the results for different values of $\alpha \in [0,1]$ (α defines the fractional degree of transformation) that approximately correspond to the apertures in Figure 7. Apart from only being gray-scale, the images from fractional Fourier transform miss the aberrations caused by the lens system. This becomes even more obvious when taking the different out-of-focus behaviour in front of and beyond the focused distance into consideration. The aperture shape in Figure 9 is horizontally compressed compared to the one in Figure 7 which is due to the lens.

Our approach can further be directly used to render lens flares by fitting the polynomials to ray traced samples that are reflected in the lens. The flare resulting from the lens shown in Figure 6 with a reflection at a cylindrical element is shown in the rendered image in Figure 10.

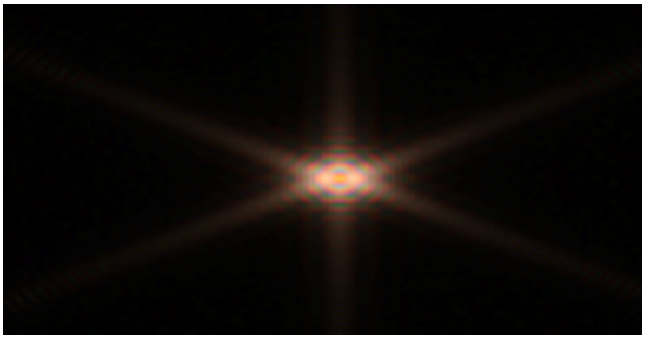
6. Limitations and Future Work

The obvious limitation of our work is that the high computation time remains relatively high as compared to simple analytical approximations. The high cost is due to the slow convergence of the Monte Carlo method in the presence of interference causing negative values in the estimator. Nevertheless, we believe that our easy to use approach resulted in a tractable algorithm.

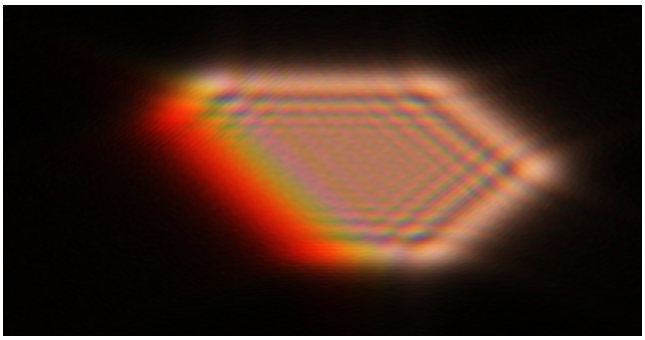
Our technique so far does not consider correlation length. We assume that all paths stay perfectly coherent no matter how far apart they are. When tracing zero-width rays, it seems difficult to consider correlation lengths. But doing so may lead to a more efficient estimator which has fewer negative contributions. We conducted some experiments with rendering lens flare and full 3D scenes instead of just point spread functions. Rendering a few point light sources already has much higher variance than rendering a single point light. This leads us to the conclusion that the source of variance is mostly the negative contributions of interference, which may make rendering point light sources one by one more efficient than rendering all at the same time. Thus, introducing a correlation length may be a generic way of gaining a lot of efficiency by discarding interference effects and cutting off many negative contributions.



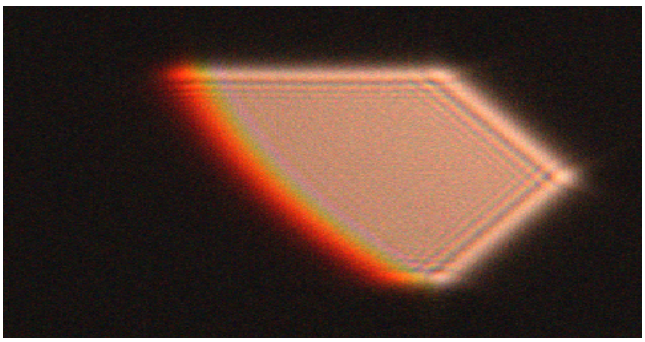
(a) $f/2$ Aperture 35×35 mm sensor



(b) $f/32$ Aperture (zoomed in)

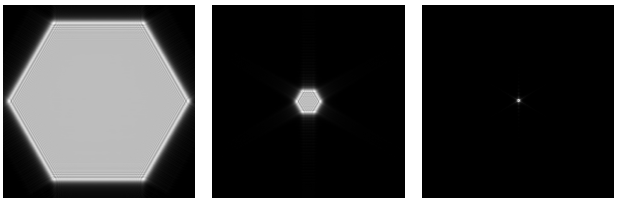


(c) $f/8$ Aperture (zoomed in)



(d) $f/2$ Aperture (zoomed in)

Figure 7: *Point spread function of an off-axis point light source rendered at different apertures. Wavelength dependent clipping at the outer pupil occurs due to rays passing the lens peripheral.*



(a) $\alpha = 0.15$ (b) $\alpha = 0.9$ (c) $\alpha = 0.99$

Figure 8: *Fractional Fourier transform can be used to approximate diffraction by a lens aperture. Lacking aberrations by the lens system, results look generally very clean and miss distortions.*

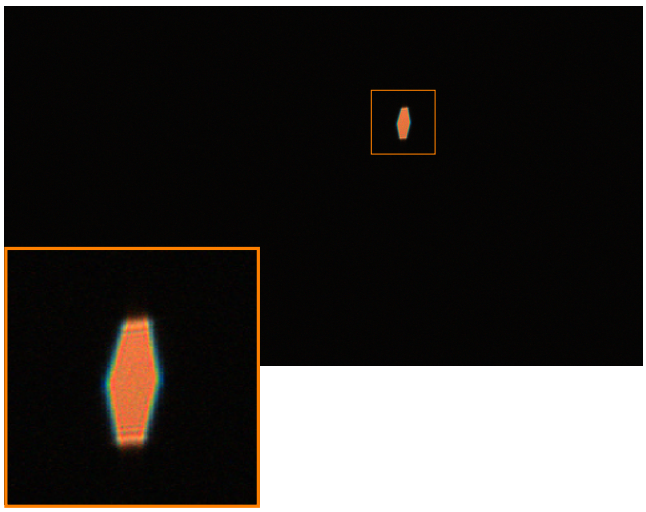


Figure 9: *The image can be focused for example through adding a sensor shift. This shift is an input of the polynomial. The above rendering shows the point spread function of an anamorphic lens. Due to cylindrical elements, the horizontal and vertical focal length differ and the image is blurred either horizontally or vertically. This depends on whether the scene point is in front or beyond the focused distance.*

In designing more efficient estimators, an open question is how to importance sample the phase space. We cannot know the full magnitude of the contribution of a path until we know all other interfering paths. One approach could be to partially replace the Monte Carlo integration by analytical approaches for special cases [SLE18]. Since this introduces a lot of assumptions (such as constant illumination), ideally this would be done in a very generic manner [Olv08].

A starting point to introduce approximations is presented by the smoothness of the phase and transmittance when plotted over aperture position. This may mean that these values would be amenable to interpolation. In the spirit of previous work [SML*12, HESL11], it may be possible to trace fewer rays and interpolate the values in between or replace the integration by piecewise analytic integration schemes.

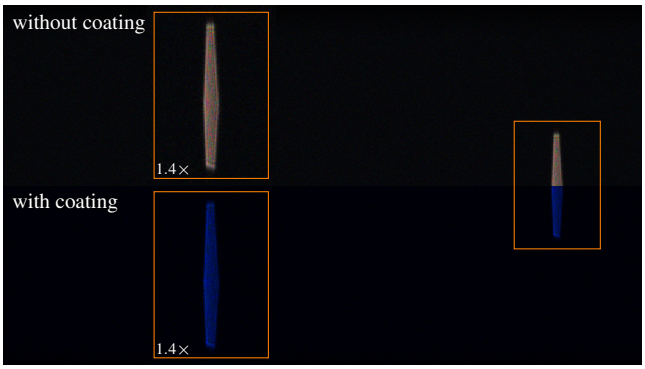


Figure 10: *Lens flares are caused by reflections inside the lens system. Our approach also works for polynomials fitted to lens flares. Here we rendered the lens flare from Figure 6, that includes a reflection at the cylindrical outer pupil. The blue color of the flare in the bottom half is caused by the anti-reflection coatings used in lenses. In the top half we removed the anti-reflection coating before fitting polynomials.*

7. Conclusion

In this work we showed that diffraction effects by a lens aperture can be simulated for point light sources even in the presence of lenses causing aberrations. Using Huygens' principle directly to integrate waves with arbitrary phase and transmittance distributions over the area of the aperture becomes viable through splitting the lens system at the aperture and approximating both parts by polynomials. Our GPU implementation is capable of accumulating 2.65 million samples per pixel per hour for a 1440×1080 frame buffer on a mid-range GPU allowing for the calculation of converged point spread functions in a few hours. We verified that Monte Carlo integration with our approach agrees with analytic approximations for special cases.

Acknowledgements

This work has been funded by the Deutsche Forschungsgemeinschaft, grant DFG 1200/4-1.

References

[Alo11] ALONSO M. A.: Wigner functions in optics: describing beams as ray bundles and pulses as particle ensembles. *Advances in Optics and Photonics 3*, 4 (Dec 2011), 272–365. 2

[BWB*99] BORN M., WOLF E., BHATIA A. B., CLEMMOW P. C., GABOR D., STOKES A. R., TAYLOR A. M., WAYMAN P. A., WILCOCK W. L.: *Principles of Optics: Electromagnetic Theory of Propagation, Interference and Diffraction of Light*, 7 ed. Cambridge University Press, 1999. 2

[HD14] HANIKA J., DACHSBACHER C.: Efficient monte carlo rendering with realistic lenses. In *Computer Graphics Forum (Proc. of Eurographics)* (2014), vol. 33, pp. 323–332. 3

[HESL11] HULLIN M., EISEMANN E., SEIDEL H.-P., LEE S.: Physically-based real-time lens flare rendering. *ACM Transactions on Graphics (Proc. SIGGRAPH) 30*, 4 (2011), 108. 3, 7

[HHH12] HULLIN M., HANIKA J., HEIDRICH W.: Polynomial optics: A construction kit for efficient ray-tracing of lens systems. In *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)* (2012), vol. 31, pp. 1375–1383. 3

[HIP15] HARVEY J. E., IRVIN R. G., PFISTERER R. N.: Modeling physical optics phenomena by complex ray tracing. *Optical Engineering 54*, 3 (2015), 1–12. 2

[JKL*16] JOO H., KWON S., LEE S., EISEMANN E., LEE S.: Efficient ray tracing through aspheric lenses and imperfect bokeh synthesis. In *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)* (2016), vol. 35, pp. 99–105. 3

[Kel62] KELLER J. B.: Geometrical theory of diffraction. *Journal of the Optical Society of America 52*, 2 (Feb 1962), 116–130. 2

[KMH95] KOLB C., MITCHELL D., HANRAHAN P.: A realistic camera model for computer graphics. In *Proc. SIGGRAPH* (1995), pp. 317–324. 3

[MWB*16] MOUT M., WICK M., BOCIORT F., PETSCHULAT J., URBACH P.: Simulating multiple diffraction in imaging systems using a path integration method. *Applied optics 55*, 14 (2016), 3847–3853. 2

[Olv08] OLVER S.: *Numerical approximation of highly oscillatory integrals*. PhD thesis, University of Cambridge, 2008. 7

[PF94] PELLAT-FINET P.: Fresnel diffraction and the fractional-order fourier transform. *Optics Letters 19*, 18 (Sep 1994), 1388–1390. 3

[SDHL11] STEINERT B., DAMMERTZ H., HANIKA J., LENSCH H.: General spectral camera lens simulation. *Computer Graphics Forum 30*, 6 (2011), 1643–1654. 3

[SHD16] SCHRADE E., HANIKA J., DACHSBACHER C.: Sparse high-degree polynomials for wide-angle lenses. In *Computer Graphics Forum (Proc. Eurographics Symposium on Rendering)* (2016), vol. 35, pp. 89–97. 1, 3, 4, 5

[SLE18] SCANDOLO L., LEE S., EISEMANN E.: Quad-based fourier transform for efficient diffraction synthesis. In *Computer Graphics Forum* (2018), vol. 37, pp. 167–176. 3, 7

[SML*12] SADEGHI I., MUNOZ A., LAVEN P., JAROSZ W., SERON F., GUTIERREZ D., JENSEN H. W.: Physically-based simulation of rainbows. *ACM Transactions on Graphics 31*, 1 (2012), 3. 7

[Sta99] STAM J.: Diffraction shaders. In *Proc. SIGGRAPH* (1999), pp. 101–110. 2

[YHW*18] YAN L.-Q., HAŠAN M., WALTER B., MARSCHNER S., RAMAMOORTHI R.: Rendering specular microgeometry with wave optics. *ACM Transactions on Graphics 37*, 4 (2018), 75. 3